AFRL-IF-RS-TR-2006-316
Final Technical Report
October 2006

METHODS, KNOWLEDGE SUPPORT, AND EXPERIMENTAL TOOLS FOR MODELING

Molecular Science Institute

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. M292/00

STINFO FINAL REPORT

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

AFRL-IF-RS-TR-2006-316 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
DANIEL J. BURNS Work Unit Manager	JAMES A. COLLINS Deputy Chief, Advanced Computing Division Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
OCT 2006	Final	Sep 01 – Sep 05

4. TITLE AND SUBTITLE
METHODS, KNOWLEDGE SUPPORT, AND EXPERIMENTAL TOOLS FOR MODELING

5a. CONTRACT NUMBER
F30602-01-2-0565

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
61101E

6. AUTHOR(S)
Roger Brent, Larry Lok

5d. PROJECT NUMBER
BIOC

5e. TASK NUMBER
M2

5f. WORK UNIT NUMBER
92

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Molecular Science Institute
2168 Shattuck Avenue
Berkeley California 94704

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

AFRL/IFTC
525 Brooks Road
Rome New York 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2006-316

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA#06-731

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Our goal was to provide software and experimental abilities to support quantitative modeling of eukaryotic systems and thus to help enable new kinds of molecular logic. The most significant accomplishments were the development of two programs. One was a "collaborative annotation", MONOD. MONOD embodied a number of genuinely novel ideas, particularly in the data structure, which constituted a middle ground between the highly structured relations of objects in a relational database and the unstructured representation of human text-based discourse and notes. The second, Moleculizer, addressed an important problem in simulating chemical reaction networks, the proliferation of closely related species of molecular complexes. This rule-based approach has since been widely adopted. Since the project period, the two programs have had different fates. Despite the notional advantages of MONOD, for the main purpose of knowledge support, we and others have opted for the free form text embodiment in a wiki (at www.openwetware.org). By contrast, work on Moleculizer continues at MSI with money from the Japanese E-cell project, and we have worked with a number of groups to "port" the basic concepts to other simulation software.

15. SUBJECT TERMS
Biological modeling, systems biology, semi-structured data storage and retrieval, collaborative annotation, chemical reaction network simulation, rule-based simulation, stochastic simulation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Daniel J. Burns
U	U	U	UL	20	19b. TELEPHONE NUMBER (*Include area code*)

Table of Contents

List of Figures

Summary

Project goals:

The goals of the original three year project were generally to develop and experimentally validate rule based software and models of a number of common "modules" of biological function. We chose the modules to represent biological processes that would be useful in building intracellular logic devices based on principles beyond "protein and DNA" logic. Task 1 developed Model Kernel codes for modeling reaction kinetics, tracking individual molecules, and representing the behavior of prototype subcomponents such as G-proteins and receptors, protein association/dissociation, kinase cascades, and induction/repression of eukaryotic gene expression. Task 2 developed experiment methods for validating various aspects of some of the intracellular functions to be modeled in the first task. Added in the fourth year, Task 3 developed codes for organization and use of supporting knowledge from protein structure databases and notes taken from natural language literature. Task 4 developed codes for simulating systems of biochemical reactions and tracking movement of large numbers of individual protein molecules within cellular space.

Accomplishments:

Significant accomplishments under this project included the development of *Monod*, a knowledge-support software tool that represented data in a structure supporting attributes of both highly organized databases and free text, supported reaction based quantitative models by enabling objects including molecular species, reactions, processes, and effects. Monod embodied fine-grained, multi-user permission controls, enabled Systems Biology Markup Language (SBML) based model import and export, included a Graphical User Interface, and an early architecture for synchronized data storage in decentralized, peer-to-peer repositories. We also developed the *Moleculizer* simulation software that provides a rule-based means to enable automatic generation of chemical reaction networks, a means to export reaction networks to other simulators, and a means to "collapse" output generated during runs into human-intelligible form. We also developed real-time fluorescence imaging methods for observing and quantifying signaling pathway protein translocation and activation events in samples of reporter strains containing various numbers of cells. Such methods provide insight into signaling pathway dynamics and a means for quantitatively measuring reaction information from small numbers of cells. The personnel contributing to this effort participated in various working groups and use cases worked throughout the program. Finally, the work under this project was described in a number of publications and lectures given by the principle investigator and contributors to this project.

Recommended future research directions:

We recommend continued development of more structured data representations and fine-grained permission control compatible with wikis, promulgation of the principle of "simultaneous SBML translation" in future knowledge support archives for biological modeling funded by the US Government, and continued development of SBML and rule-based simulation methods to handle protein complexes and better represent space.

1. Accomplishments

1.1. Monod knowledge support software

1.1.1. Represented data in structure supporting attributes of both highly organized database and free text

MONOD (in the current version, 1.5) is an interactive web application: it runs on a server, and users interact with it through a standard web browser (Figure 1). Data are stored in a conventional relational database. MONOD makes use of numerous open source software products to map data to object-oriented data structures for the programmer and to present it to the user through the web interface. The scope of an individual MONOD installation can vary from one that serves a single user to one that serves a research community. The first implementation and populated database focuses on the signal transduction pathway that governs the response of budding yeast (*Saccharomyces cerevisiae*) to mating pheromone. The early steps of this pathway are effected by biochemical reactions among a small number of proteins. The database includes information about molecular species and interactions between them (such as binding, dissociation, and post-translational modification). This structure closely matches the natural language descriptions used by biologists to describe intracellular signal transduction pathways. Data representation is "fine grained": particles of entered information can be quite small, but multiple pieces of data are easily linked together.

In contrast to the large, slowly updated blocks of information contained in individual journal papers and books, MONOD represents information as a large number of smaller connected pieces, each of which can be independently updated (Soergel, 1988, Soergel, 1977, Bush, 1945). In Eric Raymond's metaphor, we might think of books and journal articles as "cathedrals" and MONOD as a "bazaar" (Raymond, 1999). MONOD is in this sense like a Wiki, a system for collaboratively editing a set of interlinked web pages (Leuf and Cunningham, 2001) (see also http://www.wiki.org and http://www.wikipedia.org for a large-scale example), with the distinction that MONOD benefits from an organizational structure specific to its problem domain of molecular biology. While information in MONOD is often rooted in primary literature, its fine-grained data representation makes it easy to browse and to search, and these attributes may make biological knowledge more accessible to those not comfortable reading the textbooks and primary literature (for example, students and people coming from engineering backgrounds). But we note that nothing in the fine-grained data representation prevents future investigators from selecting a set of linked results, submitting those to future "journal editors" for peer review, "publishing" the linked results to the database, and having a higher value ascribed to these bodies of work.

We wrote MONOD in Java 1.4 using a number of well-known and well-tested open source software entities: the PostgreSQL relational database management system (http://www.postgres.org), the Apache web server (http://www.apache.org), the Resin-EE Java application server (http://www.caucho.com), Enterprise Java Beans (EJBs)

(http://java.sun.com/products/ejb), Xdoclet (http://www.xdoclet.org), and the eXtensible Stylesheet Language Transformations (XSLT) (http://www.w3.org/TR/xslt). In MONOD, these software entities work together to translate web-based activities into commands that store and retrieve information in the relational database. All components of the system are freely downloadable and will run on Linux, Mac OS X, and Windows. The MONOD Desktop GUI is written in Java Swing, and can be started directly from a web page using the Java Web Start technology (http://java.sun.com/products/javawebstart). It communicates with the MONOD server using Resin's Hessian binary RPC protocol (http://www.caucho.com/hessian).

We show the schema of the database that now underpins MONOD in the entity-relationship diagram in Figure 1. The most important point about this schema is that it is adapted to descriptions of biological systems that can be reduced to named molecular species and the reactions they undergo. At the same time, because many generic functions are associated with a "Coreobject" table from which all others inherit, we and others can extended this schema to include different kinds of biological knowledge. If, for example, we were to add a "Sequence" table (also inheriting from "Coreobject") to represent nucleic acid sequences, then sequence records would automatically support all of the generic functions: textual annotations, citations, user permissions, and so on. MONOD was first released in April 2002, under the GNU Lesser General Public License (LGPL) (Free Software Foundation, 1991). Its source code may be freely downloaded (from http://monod.molsci.org) and modified. As of this writing, the current version is 1.5.

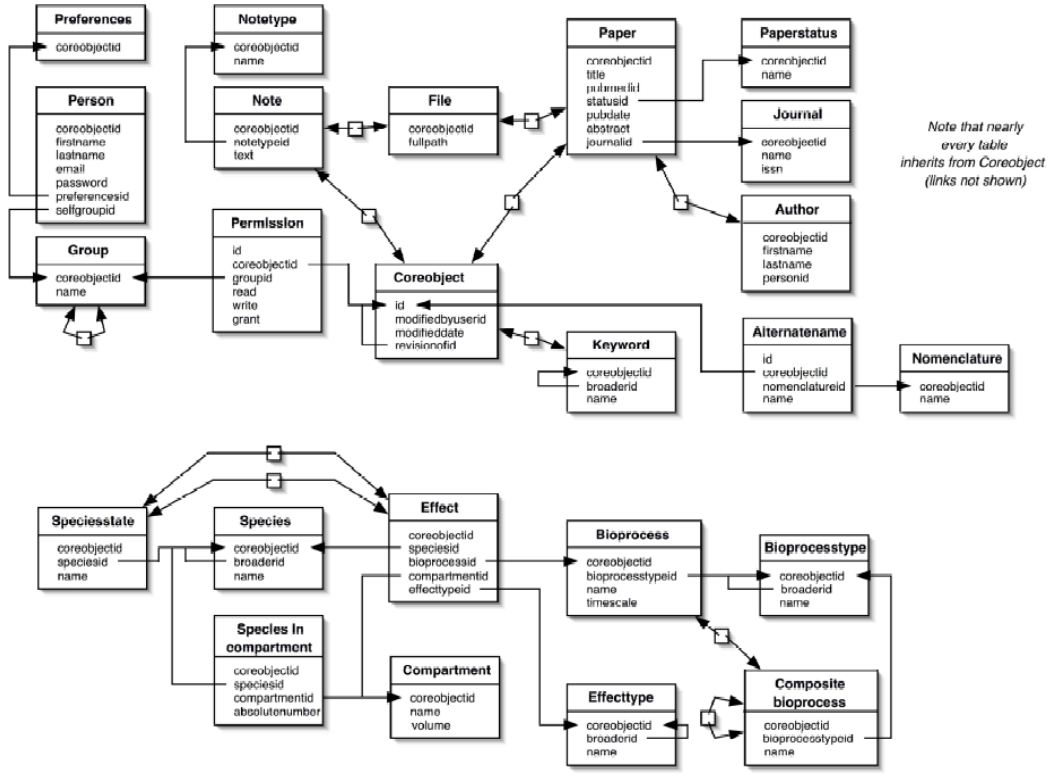

Figure 1: The version 1.5 database schema.

Core objects provide generic functionality, such as user permissions, revision control, annotations (Notes), literature citations (Papers), and keywords. Most other tables, such as Species and Bioprogress, inherit this functionality. Small squares denote many-to-many link tables, and plain arrows denote many-to-one relationships.

The thinking and progress on MONOD was described in a PDF version of a manuscript, Soergel et al., 2004, which we delivered to the project integrators. We also submitted it for publication in PLOS Biology. The paper was rejected and we have no plans to publish it at this time.

1.1.2. Supported reaction based quantitative models by enabling objects including molecular species, reactions, processes, and effects

MONOD uses a general representation for reactions and other processes. A *process* consists of a set of *effects*, where each effect describes the participation of one molecule of a species in the process and the role it plays (i.e., input, output, or catalyst), taking into account the modification state of the molecule and its location within the cell. This structure provides a consistent framework to represent different kinds of intracellular processes, including enzymatic reactions, protein complex formation, passive and active transport processes, and diffusion. For example, in MONOD a hypothetical dimerization reaction $A + B \rightarrow C$ is a process with three effects: it removes one molecule of A (from the plasma membrane, say, and only if it is phosphorylated), and removes one molecule

of B (from the cytosol), and ejects one molecule of C (into the plasma membrane, and in a certain modification state). Similarly, a hypothetical nuclear export process has four effects: it removes a molecule from the nucleus, places it in the cytosol, and hydrolyses ATP in the process, and this occurs only if a transport protein is present in the nuclear membrane (a requirement represented as a fourth effect).

To every reaction, species, state, or other object, the user can attach annotations, including citations and keywords. This capability is quite general: annotations can consist of text and attached files (such as images, movies, data tables, or other file types), annotations can be searched, and annotations can be annotated. For instance, a user might want to explain how an estimate for the number of molecules of a certain protein in an average cell was derived. The detail page for that estimate will show the annotation explaining the underlying experiments or reasoning, along with any supporting citations or graphics. Other users might add competing estimates for the same value, each with its own distinct annotations and citations. In the specific case of citations, MONOD automatically imports full references from PubMed, given a PubMed ID or journal, volume and page. The user can also select references to be imported using the integrated PubMed search tool. MONOD will download PDF versions of the selected papers when available, gaining access through the user's electronic journal subscriptions as necessary. In the specific case of keywords, for example "journal club" or "G protein", the user can access the "keyword detail" page to browse journal club papers, or entries involving G proteins. Once data and annotations have been entered, users can search the database using a standard text search box, and browse the database contents by clicking on links between records.

1.1.3. Embodied fine-grained permission control

MONOD provides a medium for structured communication among its users. The idea is that, by allowing users to make entries into the system visible to others, the program allows investigators to construct models collaboratively and to discuss them. For example, suppose researchers disagree on the value of a reaction rate; in this case, the disagreement will be recorded, and will become apparent when browsing the accumulated entries in MONOD. In this aspect, the program can also be used as a typical web discussion forum, allowing users to reply to one another's postings.

But this aspect of the program's functionality is aided by a fine-grained privilege system. Access to such discussions (indeed, to any data or annotations in the system) can be restricted to specific groups of users through this system. The program requires users to log in with a username and password, and tracks this information throughout each session. When creating or editing a record, the user can specify which individuals or groups may view the entry, modify it, or grant privileges on it to others. For example, a user might enter an idea as a private annotation. She might subsequently release it to designated individuals and, still later, to all users of that instance of MONOD, thereby "publishing" the information within that microcosm. MONOD also incorporates a revision control system, similar in concept to the Concurrent Versions System (CVS) (Cedarqvist et al., 1993) that is widely used to coordinate the development of software

projects. Like CVS, MONOD retains every revision of every record, along with a time and date stamp and the name of the user who made the revision. Normally, users only see the most current revisions of records, but they may choose to view any record from any time in the past. The revision control system allows the program to capture disagreements, and allows users to explore the history of such disagreements by studying branches of the revision tree. As researchers modify different records over time, this aspect of the program will provide a primary record of how the understanding of a biological system develops.

1.1.4. Enabled SBML based model import and export

Currently MONOD allows for import and export of data via SBML. We wish to allow the export of MONOD models to the Moleculizer stochastic reaction network generator and simulator and to other rule based simulation programs. For these purposes, we worked with the SBML group to bring about modifications SBML Level 2 to better support protein complexes and other species types (this work continues to this day).

1.1.5. Developed Graphical User Interface

We initially implemented MONOD as a web application because doing so allowed remote users to access it easily via standard browsers. It is fair to describe this first interface as a GUI. But we found that the web interface is cumbersome; many mouse clicks are required to accomplish any given task, and the resulting delays are frustrating to the user. The functionality this type of interface can offer is inherently limited. For this reason, we began developing a desktop graphical user interface (GUI) client, called MONOD Desktop (a beta version is now available). This client communicates with a backing MONOD server, but provides a more fluid and dynamic user interface (Mandel, 1997, Raskin, 2000), with contextual pop-up menus, drag-and-drop capabilities, and better navigation. MONOD Desktop includes four primary components: a search interface; an annotation editor, allowing connection of a single annotation to multiple species or processes by drag-and-drop; a diagrammatic model editor using the Kohn molecular interaction notation (Kohn, 1999, Kohn, 2001); and a “workspace navigator” for quick access to favorite items and work in progress (Figure 2).

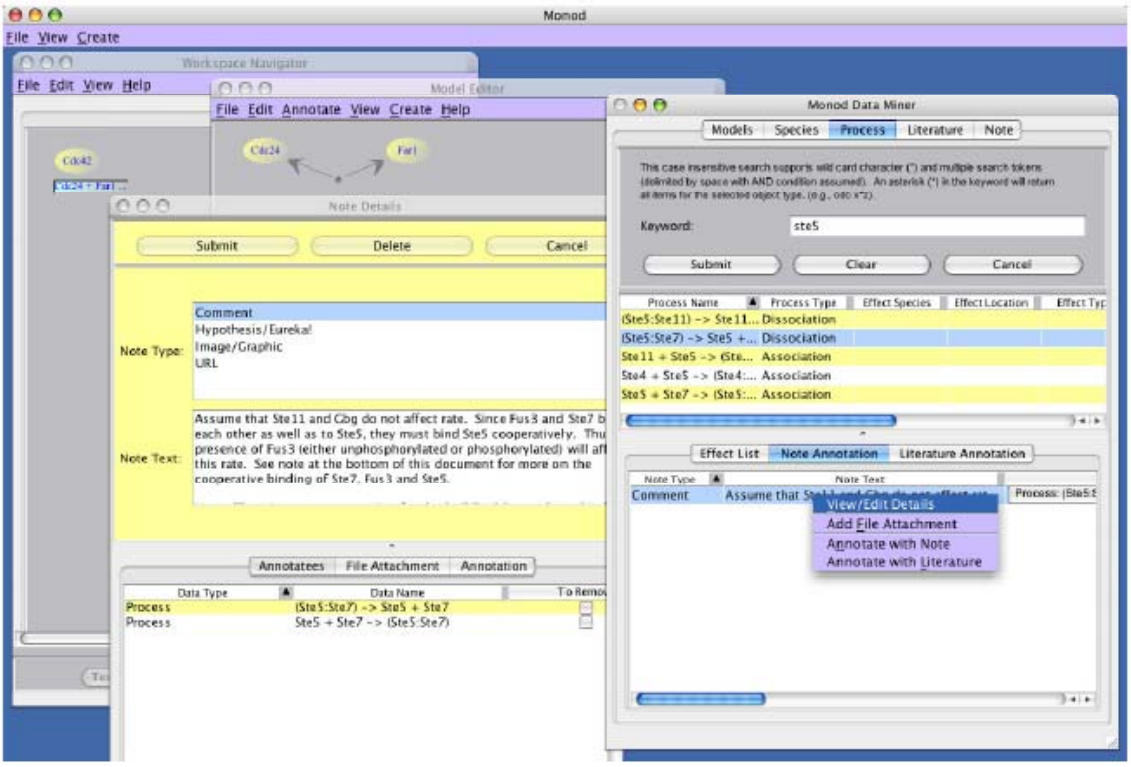

Figure 2. A view of the MONOD desktop, a graphical user interface for working with information on MONOD servers.

It allows searching, browsing, and editing data with more fluidity and ease than is possible through the web interface, and it enables users to draw diagrams of reaction networks.

1.1.6. Made progress toward architecture for synchronized data storage in decentralized, peer-to-peer repositories

In the present version of MONOD, collaboration is possible only between users of the same instance of the program, and remote users need to access the single server that runs it. If MONOD proves to be a useful knowledge sharing mechanism and the number of people using it increases, we could imagine creating a single, central, MONOD server. However, at present we believe that a linked network of distributed servers will be more secure, faster, and more reliable. In this vision, a future MONOD network would grow organically as more labs establish and maintain individual servers. This development path should better address security concerns, since private data can be stored on a local server under the physical control of laboratory that generates it, thereby ensuring local control of who accesses the data (for example, any users outside the lab group might be prevented from accessing certain data, perhaps with an exception for one trusted collaborator who backs up the data on a remote server). This architecture should also give better performance, since users would interact primarily with their local servers, which will intelligently cache remote data. In this architecture, an individual interacting with a local server would have the illusion of accessing a single worldwide database.

Toward the very end of the project period, a software developer, Jay Doane, made significant progress in working out the problems associated with this decentralized data storage, problems which are significant in any peer-to-peer computing network.

1.2. Moleculizer simulation software

1.2.1. Developed rule based means to enable automatic generation of chemical reaction networks.

Moleculizer generates reaction networks by a cyclic process (Figure 3) attached to, but largely independent of the core stochastic simulation machinery that generates reaction events. This fact makes it possible to port *Moleculizer's* reaction network generation method to stochastic simulators of other kinds.

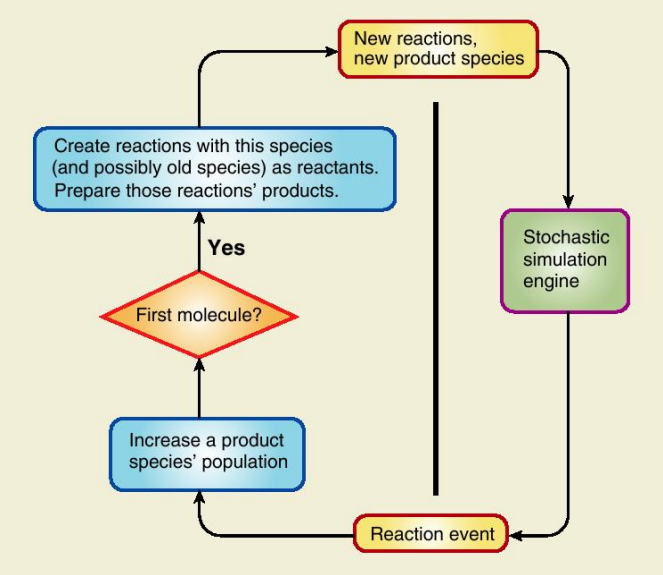

Figure 3. Reaction network generation cycle.

Moleculizer creates reactions involving a new species when the first molecule of the new species appears. If the new reactions have new product species, it enters them into a growing database of species known to the simulation. Later, when the first molecule of one of these new product species appears because a reaction event occurs, *Moleculizer* triggers the reaction generation cycle again. The bold vertical line between reaction generation and the stochastic simulation engine is intended to indicate that *Moleculizer's* species and reaction generation machinery and the basic stochastic simulation machinery are not deeply intertwined, so that *Moleculizer's* species and reaction generation machinery can be coupled easily to other kinds of stochastic simulation algorithms.

We describe how *Moleculizer* builds a reaction network with an example, the generation of a family of dimerization reactions and their products, illustrated in Figure 4. Reaction generation starts when the first molecule of a species appears in the run. The triggering molecule may appear in the initial population of the simulation or when some reaction produces it. Suppose that the molecule is a complex *C1*, and that this complex

contains a simple protein *P1*. Suppose that there is already a known complex *C2* containing another simple protein *P2*. Also, suppose that the user has specified on-rates and off-rates for *P1* and *P2* at binding sites exposed in the complexes *C1* and *C2*. *Moleculizer* asserts that the dimerization between *P1* and *P2* implies a dimerization between *C1* and *C2*, since these two complexes expose "compatible" binding sites. *Moleculizer* constructs the asserted reaction in two steps, estimating the dimerization rate, then preparing the dimerization product species *C*.

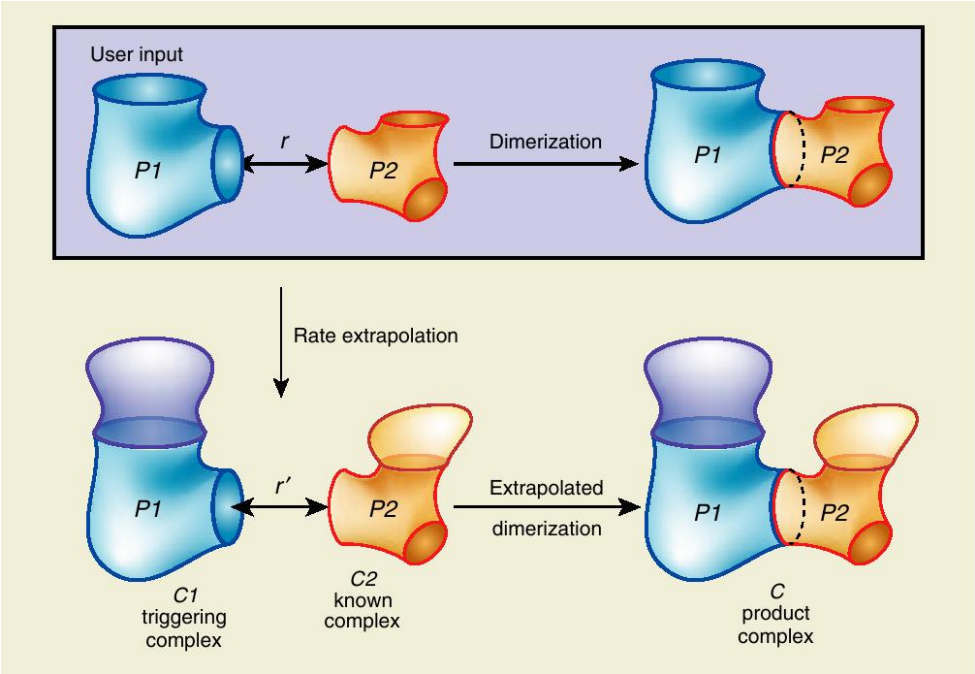

Figure 4. Dimerization example.

When the first molecule of a new complex species C1 appears, *Moleculizer* creates dimerization reactions for free binding sites exposed by C1 and free binding sites on already-known complexes such as C2 that display a compatible binding site. It extrapolates the rate of the new dimerization reaction from the rate of a user-provided prototype P1-P2 dimerization by correcting for the molecular weights of the new reactants C1 and C2. It enters the product complex C into its database of complex species when it constructs the new dimerization reaction

Moleculizer estimates the reaction rate by correcting the rate at which the simple proteins *P1* and *P2* dimerize for the larger molecular weights of the complexes *C1* and *C2*. This correction is done by reference to the formula

$$c_\mu = V^{-1}\pi d_{12}^2 (8kT/\pi m_{12})^{1/2} \exp(-u_\mu^*/kT) \qquad \text{Equation 1.}$$

from Gillespie's original exposition of the Stochastic Simulation Algorithm in (Gillespie 1976, sec.2, eq. 6) and treated further in (Gillespie 1992). This expression relates a binary reaction rate to external factors such as temperature and to physical properties of the two reacting molecules, including their masses. The masses appear in the factor $m_{12}^{1/2}$, defined by

$$m_{12}^{1/2} = \sqrt{\frac{m_1 m_2}{m_1 + m_2}}, \qquad \text{Equation 2.}$$

where m_1 and m_2 are the molecular weights of *P1* and *P2*. *Moleculizer* estimates the dimerization rate r' between the complexes *C1* and *C2*, of mass m_1' and m_2' respectively, by assuming that

$$r'\sqrt{\frac{m_1 m_2}{m_1 + m_2}} = r\sqrt{\frac{m_1' m_2'}{m_1' + m_2'}}. \qquad \text{Equation 3.}$$

This amounts to assuming that the other factors in Gillespie's formula above remain the same for the new reaction. We realize that this assumption is unwarranted for the ideal molecular diameters involved in d_{12}. In fact, *Moleculizer 1.0* does not represent or use the geometry of molecules at all, an issue we intend to address in future development.

The second step in building the new reaction is "preparing" the dimerization product species *C*. This means making an entry for *C* in the growing database of all species and their numbers known to the simulation. The program forms a two-part description of *C*, giving its structure and the states of its simple protein constituents. The structure is derived from the structures of *C1* and *C2*, and the states are the same as they were in *C1* and *C2*. If *C* has already appeared in the simulation, the program will locate it in the database of species. If *C* is new, then *Moleculizer* enters it into the database with a population of zero. But *Moleculizer* does *not* generate all the new reactions having *C* as a reactant until the first triggering molecule of *C* appears, for example, because the just-constructed dimerization reaction of *C1* and *C2* occurs for the first time. If Moleculizer did not temporize in this way, the network of all possible reactions and reactants would be generated at the start of the simulation. Instead, it generates reactions at the last instant before the simulation might demand them. By analogy with industrial production, we call this "just in time" reaction generation.

The above description of dimerization reaction construction illustrates how the program builds all the automatically generated reactions and their product species. *Moleculizer* modules provide reaction generators to construct several different classes of reactions involving complex substrates, such as dimerizations, decompositions, and enzyme-substrate reactions, along with their complex product species. This rule-based, "just in time" approach was the main goal of this DARPA-funded work.

We delivered a working version of Moleculizer and documentation to the project integrator and published Moleculizer in a well regarded article in 2005 (Lok L, Brent R. Automatic generation of cellular reaction networks with Moleculizer 1.0. Nature Biotechnology 23, 131-136 (2005)). We worked with a number of groups including Steve Plimpton at Sandia Labs to "port" the basic concepts to other simulation software. Work on Moleculizer and related simulation continues, with money from the US National Institute of Health and from the Japanese E-cell project.

1.2.2. Developed means to export reaction networks to other simulators

The appearance of Systems Biology Markup Language (SBML, http://www.sbml.org), and the development of MONOD at The Molecular Sciences Institute, both encouraged us to adopt an XML-based approach to file formats. Given the additional incentive of powerful translation facilities, such as XSLT (Tidwell 2001), we settled on XML as a "base" language for communication with and among all the programs in the Moleculizer family.

The decision to use XML as the base language necessitates an editor to help users cope with XML's verbosity and complexities of the simulation specification. We chose the Java-coded XML editor *xmloperator* (Demany, D. http://www.xmloperator.net), which runs on many platforms. *Xmloperator* provides "guided editing," which disallows changes not conforming to the specified syntax of the input file and automatically inserts required material. We wrote syntax descriptions that customize *xmloperator* to each of the file formats connected with *Moleculizer*. We wrote translators enabling *xmloperator* to convert *Moleculizer* documents into web pages linked to documentation. A user commencing to write a *Moleculizer* model is thus presented with a template document, help in filling it out, and web-based documentation to explain it.

We have provided several tools to convert *Moleculizer's* reaction network output into formats useful to other simulators. One translator generates input for *rk4tau*, an experimental stochastic simulator. *rk4tau* is based on Gillespie's "tau-leaping" (Gillespie 2001) idea. It contains parts of a standard high-order adaptive Runge-Kutta solver for ordinary differential equations. *rk4tau* is still experimental; it succumbs frequently to the same stiffness phenomenon (Deuflhard & Bornemann 2002) that hinders the use of many standard ("explicit") methods of solving ODEs when applied to chemical reaction systems. We have released it along with *Moleculizer* as a demonstration target simulator, and we anticipate that we will apply recent improvements in tau-leaping approach (Gillespie & Petzold 2003) (Rathinam *et al.* 2003) in later releases.

Another translator converts *Moleculizer's* reaction network output into input for *odie*, a simple simulator based on solving ODEs. This program uses the Bulirsch-Stoer algorithm, an "implicit" method of ODE solution that does not suffer from stiffness.

Finally, a third translator converts *Moleculizer*-generated reaction networks into SBML Level 2, a markup language to facilitate communication among biological simulation tools. Since SBML Level 2 does not handle complexes, it is necessary to refer back to the *Moleculizer* reaction network to get the structure of complex species put into the SBML Level 2 file. The next version of SBML, Level 3, will convey nearly all of the content of a *Moleculizer*-generated reaction network, including the structures of complex species and modifications of their constituents.

1.2.3. Developed means to "collapse" output generated during runs into human-intelligible form

Moleculizer allows the researcher to bundle output about elementary reactions and species into the same "biological" level of abstraction as the input. The level of abstraction is defined by the researcher. For example, a biologist can easily arrange that a single trace on an output plot give the total population of all those species of complexes that contain a particular protein. For the researcher, Moleculizer's parallel simplifications in simulation setup and output provide protection from the full, unintelligible blast of the explosion of species and reactions that appear during a large simulation.

2. Experimental validation methods to quantify biological events.

2.1. Single cell reporter strains and methods

We used an engineered "early" real-time single-cell reporter strain expressing the pathway protein Ste5 as a YFP fluorescent fusion protein to quantify Ste5-YFP movement from the cytoplasm to the plasma membrane in response to the addition of an external signal, alpha factor. From such experiments we hoped to be able to produce a measurement of the binding constant of Ste5 to another pathway protein, Ste4 at the membrane, as well as a measure of the diffusion constant of Ste5 in the cytoplasm. Progress also included the development of a statistic that is sensitive to translocation, as well as a model to account for the observed data.

2.2 Mass Spectrometry Methods

Progress towards developing mass spectrometric methods to measure the information processing components from a very small numbers of cells. Though this work was high-risk, we made progress developing experimental methods to make sensitive, quantitative measurements of post-translationally modified proteins from thousands of yeast cells, with the goal of optimizing these methods to increasingly smaller numbers of cells. Preliminary experiments show that there are significantly more sites of phosphorylation on pathway proteins than have previously been reported in the literature. We are now systematically cataloging sites of phosphorylation on all pathway components in the presence and absence of alpha factor.

2.3 Flourescence and Cell Tracking Methods

We also to analyzed transcriptionally-activated fluorescence reporters in single cells, with a goal of developing software to measure several quantitative characteristics of the alpha factor system. We created Cell-ID 1.0, a cell-tracking code and data analysis program for use with images from single cells on a fluorescence microscope. This advance enables a

user to take several hundred well-focused bright-field/fluorescence data samples without human oversight. We have demonstrated that with Cell-ID 1.0 the movement of a protein from the cytoplasm to the membrane, the association and dissociation of pairs of nuclear proteins, and quantitative differences between individual cells can be studied using fluorescent reporter molecules.

We also developed experimental methods to make sensitive, quantitative measurements of post-translationally modified proteins from thousands of yeast cells, with the goal of optimizing these methods to increasingly smaller numbers of cells. Experiments showed that there are significantly more sites of phosphorylation and ubiquitinylation on pathway proteins than have previously been reported in the literature.

We used the Odyssey Infrared Imaging System (Li-Cor) to quantitatively measure pathway proteins from small numbers of yeast cells by Western blot analysis. Analysis to date on 13 of 25 pathway proteins shows that there are large differences in the number of pathway components per cell. The range determined varies from several hundred molecules per cell to 20,000 molecules per cell. Both the methods for quantification and the most striking initial conclusion arising from them, is that the number of molecules in many cases varies greatly from published results, are relevant to the work of many biologists.

3. Recommended future research directions

3.1. Continued development of more structured data representations and fine grained permission control compatible with wikis.

The idea that no biological model should ever be distributed without the ability for any user to learn about the sources of information used and choices made by the model-builder seems so logical that we wish every member of the model-building community would practice it. We believe that DARPA, NIH, and NSF should make doing so a condition of future government funded work. To that end, we are articulating that as a "pillar" of "principled model development" in a forthcoming manuscript by Kirsten Benjamin *et al.* and promulgating it at openwetware.org. Also, we are communicating with members of the wiki community to try to make fine-grained permission control and revision tracking a part of future wiki development. We are also frequently involved in discussions as to how one might make storage and retrieval of knowledge in wikis more structured, while recognizing that, at the core, this is a terribly difficult problem given how wikis work.

3.2. Promulgation of the principle of "simultaneous SBML translation" in future knowledge support archives for biological modeling funded by the US Government.

We advocate the principle that both the documentation for a model and the (differential) equations or chemical reaction networks that represent it should be immediately translatable into SBML. Different ways of representing the model are thus put on an equal, though differently computable, footing. We are recommending that US Government funded research requires this as a condition for funding work on models of chemical reaction networks. And we are promulgating this as another "pillar" of principled modeling, both at www.openwetware.org and in the forthcoming Benjamin *et al.* paper.

3.3. Continued development of SBML and on rule-based simulation methods to handle protein complexes and better represent space.

We continue to work with the SMBL organization to ensure that SBML develops in ways that are compatible with our needs in work with models of intracellular information processing systems. The issues come down as always to protein complexes, and, increasingly, to means to represent space, via large intracellular compartments, via smaller cells (here meaning "mesh elements" or "voxels,") or via tracking the movement of individual molecules. For SMBL interoperability regarding complexes and spatial simulation, we maintain direct contact with SBML developers. Similarly, we work with the *E-Cell* project, an international, Japanese-funded, open source software development project in our development of new ways of handling complex species and spatial simulation. For simulations at the Molecular Sciences Institute, we have developed a spatial (compartment-based) version of Moleculizer, and we are developing a molecule-tracking version in conjunction with *E-Cell* and with the *ChemCell* project at Sandia National Laboratory. We are actively engaged in porting *Moleculizer* concepts to other simulators compatible with E-cell.

4. References

Bush V (1945) As We May Think. The Atlantic Monthly 176: 101-108.

Cederqvist P, Pesch R, Grubbs D, et al. (1993) Version Management with CVS. http://www.cvshome.org

Deuflhard, P. & Bornemann, F. (2002) *Scientific Computing with Ordinary Differential Equations.* Springer-Verlag, New York, New York.

Free Software Foundation (1991, 1999) GNU Lesser General Public License. http://www.gnu.org/copyleft/lesser.html.

Gillespie, D. (1976) A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comp. Phys.* **22**, 403–434.

Gillespie, D. (1992) *Markov processes: an introduction for physical scientists.* Academic Press, Boston, Massachusetts.

Gillespie, D. (2001) Approximate accelerated stochastic simulation of chemically reacting systems. *J. Chem. Phys.* **115(4)**, 1716–1733.

Gillespie, D. & Petzold, L. (2003) Improved leap-size selection for accelerated stochastic simulation. *J. Chem. Phys.* **119(16)**, 8229–8234.

Kohn KW (1999) Molecular interaction map of the mammalian cell cycle control and DNA repair systems. Mol Biol Cell 10: 2703-2734.

Kohn KW (2001) Molecular interaction maps as information organizers and simulation guides. Chaos 11: 84-97.

Leuf B, Cunningham W (2001) The Wiki Way: Collaboration and Sharing on the Internet. Boston: Addison-Wesley

Lok, L, Brent, W (2005) Automatic generation of reaction networks with Moleculizer 1.0 Nature Biotechnology 23, 131-136

Mandel T (1997) The Elements of User Interface Design. New York: John Wiley and Sons.

Raskin J (2000) The Humane Interface: New Directions for Designing Interactive Systems. Boston: Addison-Wesley.

Rathinam, M., Petzold, L., Cao, Y. & Gillespie, D. (2003) Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *J. Chem. Phys.* **119(24)**, 12784–12794.

Raymond ES (1999) The cathedral and the bazaar. San Francisco: O'Reilly and Associates.

Soergel, David (1998) OrganISM, an abstract representation paradigm for intelligent information networks. Stanford, CA: Stanford University, Symbolic Systems Program. http://www.davidsoergel.com/organism.html.

Soergel, Dagobert (1977) An automated encyclopedia -- a solution of the information problem? International Classification 4(1): 4-10; 4(2): 81-89.

Tidwell, D. (2001) *XSLT*. O'Reilly & Associates, Inc. Sebastopol, California.

5. Publications

Lok, L, Brent, W (2005) Automatic generation of reaction networks with Moleculizer 1.0 Nature Biotechnology 23, 131-136.

Soergel, D, Choi, K., Thomson, T., Doane, J., George, B., Morgan-Linial, R., Brent, R., and Endy, D. MONOD, a collaborative tool for manipulating biological knowledge. Internal publication to DARPA biocomp 2004, submitted for publication, and to be published via openwetware.org.

6. Personnel

Benjamin, K.
Brent, R.
Choi, K.
Colman-Lerner, A.
Doane, J.
Endy, A.
Kroll, J.
Lok, L.
Morgan-Linial, R.
Resnekov, O., and Soergel, D.